Manual Técnico

OpenDomo

ÍNDICE

DESARROLLO DE SCRIPTS **1**

REQUISITOS PARA LOS SCRIPTS **1**

SINTAXIS DE LOS SCRIPTS **3**

TIPOS DE SECCIONES EN LOS SCRIPTS **3**

TIPOS DE CAMPO EN LOS SCRIPTS **7**

FORMATO DE TEXTO DE SALIDA EN LOS SCRIPTS **9**

DETALLES TÉCNICOS ADICIONALES DEL HARDWARE **20**

TOMA POLARIZADA **20**

INTRODUCCIÓN A LOS SERVOMOTORES **22**

PRINCIPIOS DE FUNCIONAMIENTO **24**

ALARMA LÁSER **26**

ÍNDICE DE CUADROS

CUADRO 1

EJEMPLO DE SCRIPT BÁSICO.....2

CUADRO 2

EJEMPLO DE UN SCRIPT PRINCIPAL 3

CUADRO 3

ESTRUCTURA BÁSICA DE UN TEMA O ESTILO.....10

CUADRO 4

CREACIÓN DE UN TEMA PARA OPENDOMO 12

CUADRO 5

CREACIÓN DE UN SKIN PARA OPENDOMO 16

CUADRO 6

CONFIGURACIÓN DE SERVOMOTORES PARA MARCAS CONOCIDAS 25

ÍNDICE DE GRÁFICOS

GRÁFICO 1

SALIDA DEL SCRIPT PRINCIPAL 3

GRÁFICO 2

SELECCIÓN DE INTERFAZ 19

GRÁFICO 3

SALIDA DE NUEVA INTERFAZ 19

GRÁFICO 4

TOMACORRIENTE POLARIZADO 20

GRÁFICO 6

TOMACORRIENTE NO POLARIZADO 21

GRÁFICO 7

INSTALACIÓN DE UN TOMACORRIENTE POLARIZADO 21

GRÁFICO 8

SERVOMOTOR DE ROTACIÓN CONTINUA 22

GRÁFICO 9

COMPONENTES DE UN SERVOMOTOR 23

GRÁFICO 10

COLORES DE LOS CABLES DE LOS PRINCIPALES FABRICANTES DE SERVOMOTORES 24

GRÁFICO 11

PULSOS PWM PARA CONTROLAR SERVOMOTORES 25

GRÁFICO 12

ESQUEMA DE LA ALARMA LASER 26

GRÁFICO 13

MONTAJE DE LA ALARMA EN UN PROTOBOARD 27

GRÁFICO 14

ESQUEMA DEL TABLERO PRINCIPAL 28

GRÁFICO 15

MONTAJE DEL TABLERO PRINCIPAL 29

GRÁFICO 16

ÁREAS DE LA MAQUETA..... 30

GRÁFICO 17

DIFERENTES PERSPECTIVAS DEL PROTOTIPO..... 31

Desarrollo de Scripts

Para desarrollar los scripts que necesitemos, debemos saber un poco de programación en Shell a continuación se detallara los requisitos que necesitan nuestros scripts para su correcto funcionamiento.

Requisitos para los scripts

En este punto se enumeran los requisitos básicos de la creación de scripts estándar. Estos scripts serán considerados como servicios exportables, de modo que el sistema distribuido de la red doméstica pueda administrarlos de forma automática.

1. **Nombre:** el nombre de los scripts deberá cumplir el formato "a-zA-Z0-9.sh" y tener un máximo de 40 letras. El nombre de los scripts no puede contener espacios, pero la primera letra de cada palabra que lo componga será mayúscula, salvo la primera palabra.
2. **Cabecera:** la cabecera del archivo deberá contener:
    - un comentario del tipo "#package: nombrepaquete" indicando el código del paquete al que corresponda,
    - un comentario del tipo "#desc: descripción de una línea" con una breve explicación del cometido del propio script y

- un comentario del tipo "#type:[local|multiple|restricted|redundant]" indicando cómo deberá ser tratado por **oddiscovery** en la propagación de los servicios en la red OpenDomo.

3. **Ubicación:** los scripts se guardarán en /usr/local/opendomo/services/ por lo que el nombre de tales scripts deberá ser suficientemente descriptivo para evitar colisiones.
4. **Robustez:** es responsabilidad de los scripts efectuar la comprobación de permisos de usuario; asimismo, tales scripts deberán tener permisos de ejecución para aquellos grupos que deban poderlos ejecutar.
5. **Idioma:** los textos de salida de los scripts, así como el contenido de la descripción en la cabecera, será en inglés simplificado, y la interfaz pertinente ya se hará cargo de las traducciones.

Cuadro 1 Ejemplo de script básico

```
#!/bin/sh
# Este script activa la bajada de una puerta de garaje a escala
Echo on > /var/opendomo/control/domino/purdn
# Esperamos 10 segundos el tiempo total
Sleep 10
# Deja de enviar la señal
Echo off  > /var/opendomo/control/domino/purdn
```

Elaboración: Ricardo Almendares

Ahora invocaremos desde un script principal “puerta.sh” a “purdn.sh”

Cuadro 2 Ejemplo de un script principal

```
#!/bin/sh
#desc: mi script
#type: local
if test “$2” == “Abrir” ; then
        /etc/opendomo/purup.sh &
        echo “$2” > /tmp/myscript.tmp
fi
if test “$2” == “Cerrar” ; then
        /etc/opendomo/purdn.sh &
        echo “$2” > /tmp/myscript.tmp
fi
```

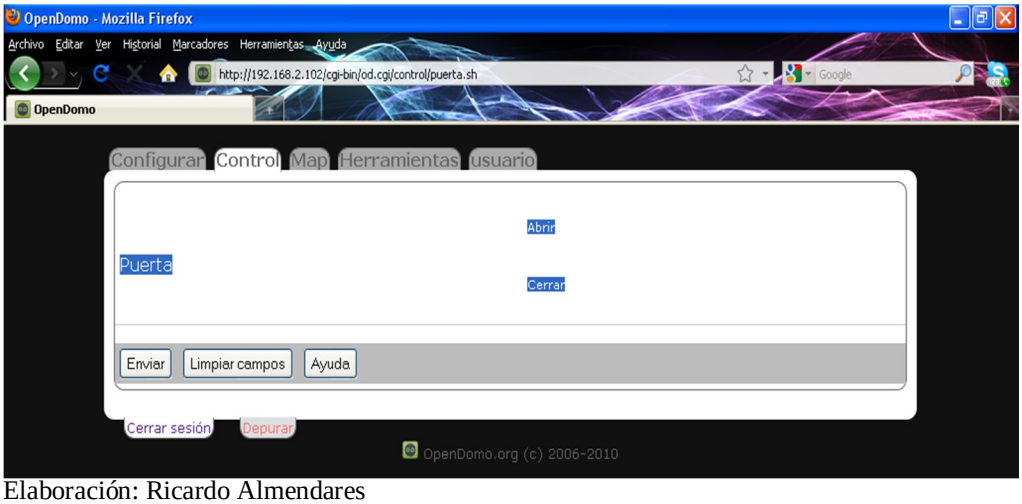
1

```
status=`cat /tmp/script.tmp` 2 > /dev/null
echo “form: `basename $0`”
echo “ action Puerta subcommand[Abrir, Cerrar] $status”
echo
```

2

Elaboración: Ricardo Almendares

Gráfico 1 Salida del script principal



Elaboración: Ricardo Almendares

Sintaxis de los scripts

Tipos de secciones en los scripts

La salida generada por los scripts puede ser formateada de forma muy diversa por parte de la interfaz (en este caso, **odcgi**), por lo que deberemos indicar qué clase de interfaz

estamos generando: no es lo mismo componer una interfaz visual adaptada para un formulario que para un listado. Para ello emplearemos las secciones.

Existen cuatro tipos de secciones: categories, usage, list y actions. Indicará el inicio de sección la propia palabra clave a principio de línea seguida de ":" (a partir de **odcgi** v0.5).

- **form**: (antiguo "usage") inicio de formulario. Los elementos que le sucedan serán formateados como campos de un formulario, cerrándose al inicio de la próxima sección, o en su defecto al final del script. Si no se ha encontrado una sección "actions", se añadirán automáticamente los botones estándar de formulario web ("Submit" y "Reset"). El envío del formulario se hará al propio script, o aquél indicado en la línea de "usage", después del caracter ":".
- **list**: inicio de listado (tipo por defecto). Los elementos que le sucedan serán mostrados como un listado tabulado, siendo enviados como siguiente parámetro al script indicado en la línea de cabecera, después del caracter ":".
- **tabs**: (antiguo "categories") caso especial de listado, empleado como cabecera de la interfaz. Los elementos serán mostrados como pestañas (en el tema por defecto) y su clave será empleada como primer parámetro al propio script.
- **actions**: los elementos que le sucedan serán mostrados como botones, enviando el contenido de la sección anterior al script indicado en la primera columna.

Las secciones quedarán delimitadas por líneas en blanco. En su defecto, quedarán encadenadas, viendo modificado su funcionamiento del siguiente modo:

- **list + actions**: los elementos de la lista que se encuentren seleccionados se pasarán como parámetros al botón de acción pulsado. Esto es especialmente útil en el caso que el botón sea "Borrar", o para modificar alguna característica de tales elementos.
- **form + actions**: los datos introducidos por el usuario en el formulario serán enviados al script correspondiente al botón pulsado.
- **list + form**: el script del listado prevalecerá, mientras que el formulario se mostrará debajo del mismo. Muy útil para construir filtros sobre los listados.

Ejemplos

La siguiente salida generará un sencillo formulario con tres campos:

```
form: sendMail.sh
      recipient    Recipient    text
      subject      Subject      text
      message      Message      longtext
```

No es necesario indicar sección "actions", por lo que al hacer el "Submit", se enviará al propio script "sendMail.sh" como primer, segundo y tercer argumento. Si queremos añadir nuevas funciones al formulario anterior, podemos emplear la versátil sección "actions".

```
form: sendMail.sh
      recipient    Recipient    text
      subject      Subject      text
      message      Message      longtext
```

```
actions:
      sendMail.sh   Send mail
      saveMail.sh   Save mail to drafts
```

Obsérvese que es necesario indicar el propio script si queremos que aparezca como una opción más. Los campos serán enviados como primer, segundo y tercer argumentos del mismo modo a ambos scripts.

En el siguiente ejemplo veremos una forma de utilización estándar de los listados, en una típica interfaz de "altas, bajas y modificaciones":

```
list: viewUsers.sh
      opalenzuela  Oriol Palenzuela    user admin
      dlerch       Daniel Lerch        user admin
      tester       Test user           user
actions:
      addUser.sh    Add new user
      viewUsers.sh  Edit user
      delUSer.sh    Delete user
```

Tal como podemos ver, el script de cabecera "viewUsers.sh" también aparece como un elemento de la sección "actions". Esto significa que se trata de la acción por defecto, y sucederá lo mismo si seleccionamos un elemento de la lista (en este caso un usuario) y pulsamos el botón "Edit user", como si hacemos click en el propio elemento: se llamará al script "viewUsers.sh" pasando como primer argumento el código del elemento seleccionado (primera columna, por ejemplo "tester").

En algún caso, para reducir el número de enlaces simbólicos creados en "/etc/opendomo/root/" podemos emplear scripts "concentradores". Podemos hacerlo muy fácilmente empleando solamente una sección "actions", de modo que se mostrará una lista de botones con textos descriptivos.

Hay algunos scripts que deberán mostrar mucha información al mismo tiempo. Para estructurarla podemos recurrir a la división de categorías, empleando la sección "categories". Esta sección es la empleada para construir la división de pestañas de la cabecera de la interfaz (Configurar, Control, Seguridad, etc.), pero también puede ser usada en el cuerpo de los scripts. Esta funcionalidad está siendo desarrollada todavía, y por ahora es considerada experimental.

Tipos de campo en los scripts

La salida generada por los scripts emplea listados y formularios (dependiendo de las secciones) cuyos elementos se caracterizan por un tipo de campo. Este tipo de campo determinará la forma en que la interfaz se compone, facilitando al usuario la interacción. Los elemento, tanto de listados como de formularios, se muestran tabulados en cuatro columnas: identificador, texto descriptivo, tipo de campo y (opcionalmente) valor complementario.

Los tipos de campo soportados por los formularios son:

- **hidden**: el elemento se conservará con su valor y será enviado junto con el resto de campos del formulario, pero no será mostrado.
- **readonly**: se mostrará el texto descriptivo y el valor, pero su contenido no podrá ser modificado por el usuario.
- **text**:[]: se mostrará el texto descriptivo y el valor del campo, en un recuadro editable por el usuario. El bloque a partir de los dos puntos (opcional) permitirá indicar una expresión regular para la validacion del contenido del campo.
- **longtext**: igual que text, pero en un recuadro de texto multilínea.
- **image**: se mostrará la imagen cuya URL se indica en el campo de valor complementario.
- **video**: igual que "image", pero empleando el tag *embed*.
- **application**: igual que "video" e "image", empleando un iframe.
- **list**[]: se mostrará un campo desplegable con las opciones indicadas dentro de los corchetes, separados por coma.
- **subcommand**[]: se mostrarán los elementos del listado como botones

Por defecto, los listados se mostrarán de forma tabulada, incorporando los tipos especificados como clases del elemento de lista, aunque se considerarán algunos tipos como excepciones. Los tipos de campo especiales soportados por los listados son:

- **file**: el elemento de lista tendrá un hiper vínculo al archivo referenciado, empleando la API del **odcgi.**
- **file.image**: igual que el caso anterior, con la particularidad de incluir la imagen al principio del elemento de lista.
- **sub**: el elemento quedará anidado por debajo del nivel del elemento anterior. Con Javascript activado se podrá colapsar y expandir independientemente (pendiente de desarrollar).

Formato de texto de salida en los scripts

Cuando los scripts llamados por la interfaz quieran mostrar una salida textual, podrán hacerlo prefijando las líneas de texto con el carácter "#". Si, además, esta salida debe estar formateada de una forma en particular, el prefijo podrá complementarse tal como se muestra en el siguiente ejemplo:

```
# Este texto se mostrará como un párrafo convencional
#WARN Este mensaje está resaltado
#ERR Este bloque se mostrará como mensaje de error
#URL http://www.opendomo.com
```

En las nuevas versiones de **odcgi** es posible emplear campos variables. Para ello será necesario delimitar la parte variable del mensaje mediante "[" y "]". Por ejemplo:

```
# El paquete [odcgi] ha sido actualizado
```

NOTA: La salida de los mensajes debe ser en inglés, ya que el CGI se encarga de la traducción.

Una vez descrito todo lo que se necesita saber para que nuestros scripts funcionen correctamente, procedemos a crear algún tema y un estilo para ello necesitamos saber un poco de programación en CSS (Cascading Style Sheets) hojas de estilo en cascada, ya que este es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML.

Cuadro 3 Estructura básica de un tema o estilo

```
/* OpenDomo CGI interface's structural style
#desc Hoja de estilo estructural
 Los estilos aquÃ definidos deben aplicarse para
cualquier tema o skin, ya que
 forman parte del comportamiento del CGI
 */

.hidden {
        display: none!important;
}
.collapsed li{
        display: none!important;
}

.detailedlist {
        width: 100%;
        list-style: none;
        display: table;
        border-collapse: collapse;
        table-layout: fixed;
}
.detailedlist li{
        display: table-row;
        list-style: none;
}
.detailedlist li p{
        display: table-cell;}
```

1

```
.detailedlist li .imgframe{
        display: table-cell;
        width: 60px;
        overflow:hidden;
        vertical-align:middle;
        text-align: center;
}
.listbox{
        width: 100%;
        list-style: none;
        display: table;
        border-collapse: collapse;
        table-layout: fixed;
}
.listbox li{
        display: table-row;
        list-style: none;}
.listbox li p{
        display: none;}
.listbox li label {
        display: table-cell;}
.listbox li .imgframe{
        display: table-cell;
        width: 60px;
        overflow:hidden;
        vertical-align:middle;
        text-align: center;}
```

2

```
.iconlist{
		list-style-type: none;
}
.iconlist li{
		display: inline-block;
		height: 90px;
		width: 90px;
		border: 1px solid #666;
		overflow: hidden;
}
.iconlist li p {
		display: none;
}
.iconlist li label{
		float:baseline;
		font-size: 10pt;
		white-space: nowrap;
}
.iconlist li .imgframe{
		padding: 2px;
		height: 60px;
		width: 60px;
		overflow:hidden;
}
.iconlist li .imgframe img{
		vertical-align:middle!important;
		text-align: center;
}
.tabform{
		width: 100%;
		list-style: none;
		display: table;
		border-collapse: collapse;
		table-layout: fixed;
}
.tabform li {
		display: table-row;
		list-style: none;
		padding-left: 5px;
}
.tabform li label, .tabform li p {
		display: table-cell;
}
```

3

```
.tabform li label{
		padding-left: 20px;
}
.tabform .separator label {
		padding-left: 5px!important;
}


.toolbar{

}
.toolbar a{
		cursor:pointer;
}
.listboxtoolbar{
		text-align:right;
		margin-right: 3px;
}
.errcont{
		background-position: center left;
		background:url(/images/warning.png);
		background-repeat:no-repeat;
		padding-left: 32px!important;
}

#advpad {
		display: block;
		margin: 15px;
		height: 200px;
		overflow: auto;
}
.nkey {
		margin: 3px;
		width: 70px;
		height: 40px;
		font-size: 110%;
}

.root, .applicationTitle{
		display:none;
}
```

4

Fuente: OpenDomo

Cuadro 4 Creación de un tema para OpenDomo

```
/* OpenDomo CGI interface's theme
#Tema creado por Ricardo Almendares
#
# Ha sido optimizado para Firefox y Safari

h1 {
        text-align: center;
        font-size: 18pt;
}
h2 {
        text-align: center;
        font-size: 12pt;
}
body {
        font-family: Verdana, Geneva, Arial,
Helvetica, sans-serif;
        width: 80%;
        margin-left: auto;
        margin-right: auto;
        margin-top: 20px;
        padding: 2px;
}
#loginform_frm {
        -moz-border-radius: 10px;
        border-radius: 10px;
        margin: 0px;
        padding: 10px;
        background-position:center;

        background-image:
url(/images/panel.png);
        background-repeat:no-repeat;
}
#loginbox li label{
        text-align:right;
}
#main {
        text-align    : left;
        }
```

5

```
.iconlist li{
        -moz-border-radius: 10px;
        -webkit-border-radius: 10px;
        border-radius: 10px;
}
a em {
        display: block;
        margin-top: 0.3em;
}
#header .categories li a{
        font-weight: bold;
}

.categories {
        list-style: none;
        border: none;
        padding: 0px;
        margin: 0px;
        margin-left: 4px;
}
.categories li {
        display: inline !important;
        margin: 0px;
        padding: 0px;
}
.categories li a {
        background-position: center left;
        background-repeat:no-repeat;
        margin: 1px;
        border: 1px solid gray;
        border-bottom-style: hidden;
        -moz-border-radius-topleft: 10px;
        -moz-border-radius-topright: 10px;
        -webkit-border-radius-topleft: 10px;
        -webkit-border-radius-topright: 10px;
        border-radius-topleft: 10px;
        border-radius-topright: 10px;
        text-decoration: none;
        cursor: pointer;
        padding: 1px;
        padding-bottom: 0px;
}
```

6

```
#footer {
	font-size: 11px;
}
#footer b{
	font-weight: none;
}
#filler {
	display: none!important;
}
input[type=button], input[type=submit],
input[type=reset], .button {
	padding: 2px;
	margin: 2px;

}
.button {
	border: 1px outset;
	text-decoration: none;
	border-radius: 4px;
	-moz-border-radius: 4px;
	-webkit-border-radius: 4px;
}
.pressed {
	border: 2px inset;
	text-decoration: none;
	background-color: #33A;
	color: white;
}

input[type=text], input[type=readonly] {
	width:100%;
}
.subtab {
	padding: 2px;
	margin: 2px;
	margin-left: 20px;
	border: 1px solid gray;
	border-top-style: hidden;
	-moz-border-radius-bottomleft: 10px;
	-moz-border-radius-bottomright: 10px;
	-webkit-border-radius-bottomleft:
10px;
```



```
-webkit-border-radius-bottomright: 10px;
	border-radius-bottomleft: 10px;
	border-radius-bottomright: 10px;
	text-decoration: none;
	font-size: 12px;
	background-color: white;
}
fieldset {
	border: 1px solid #666;
	border-radius: 10px;
	-moz-border-radius: 10px;
	-webkit-border-radius: 10px;
	margin: 0px;
	padding: 0px;
	padding-top: 5px;
	padding-bottom: 5px;
}
legend {
	margin-left: 10px;
}
/* **** T A B U L A C I O N ********
 1. La clase "tabform" se muestra como un
formulario tabulado
 2. La clase "listbox" se muestra como una lista
tabulada, cuyos elementos son
    activos
 3.
*/
.iconlist li:hover, .listbox li:hover, .detailedlist
li:hover,.focused {
	background: #DFDFDF;
	color: #222;
}
.tabform li, .listbox li, .detailedlist li{
	border-bottom: 1px solid #CCC;
	height:30px;
}

.tabform li label, .listbox li label {
	margin: 1px;
	padding-left: 5px;
	vertical-align:middle;
}
```

```
.tabform li p, .listbox li p {
		vertical-align:middle;
		font-size: 11px;
		padding-left: 5px;
}
.tabform, .listbox {
		padding: 0px;
}
.listbox li .imgframe {
		padding-left: 2px;
}
/* *** CONTROLES VISUALES ****** */
.off .pressed, .on .pressed {
		display: none;
}
.on a, .off a{
		color:transparent;
		-webkit-border-radius: 10px;
		-moz-border-radius: 10px;
		display:block;
		overflow: hidden;
	height: 30px;
		width: 30px;
		#border: 1px solid gray;
}
.on a{

   background: url(/images/on.png) no-repeat;

}
.off a{

   background: url(/images/off.png) no-repeat;
}
.gauge a{
		display:inline-block;
		border:0px;
		margin:0px;
		color:transparent;
}
```

9

```
.gauge .pressed {
   background: #fff url(gauge_pin.png) 0 0 no-
repeat;
}
.gauge .unpressed {
   background: #fff url(gauge_track.png) 0 0 no-
repeat;
}
.percent p {
		width: 100%;
		border: 1px solid gray;
}
.percent p input {
		border: 0px;
		text-align: right;
		background-color: blue;
}
/* *** T E X T O ************ */
p {
		padding: 0px;
		margin: 1px;
		font-size: 10pt;
}
.debug {
		display: none;
		border: 1px solid gray;
		margin-left: 15px;
		margin-right: 15px;
		background: #DFDFDF;
}
.message p{
		text-align: left;
}
.sub label{
		padding-left: 20px!important;
}
.unavailable {
		display: none;
}
```

10

```
.noscript {
		font-size: 12pt;
		color: #BCBCBC;
}
.toolbar {
		margin-left:260px;

		padding:2px;

}

.imgframe {
		text-align:center;
		margin-left:10px;
		margin-right:10px;
}
.hidden {
		display: none!important;
}

#loginbox{
		border-collapse: collapse;
		float: center;
}
#loginbox td{
		padding: 10px;
}
.copyright{
		margin-top: 5px;
		text-align:center;
}
.copyright a{
		text-decoration: none;
}
```

11

Elaboración: Ricardo Almendares

Nota: Las imágenes deben ser guardadas en la siguiente dirección: /var/www/images y el script debe ser colocado dentro de una carpeta en la ruta /var/www/themes

Cuadro 5 Creación de un skin para OpenDomo

```
/* OpenDomo CGI interface's skin
#Tema creado por Ricardo Almendares
#
# Ha sido optimizado para Firefox y Safari
body {
        font-family: Verdana, Geneva, Arial,
Helvetica, sans-serif;

        color: #222;
        background-image:
url(/images/wood1_by_yc.jpg);
        background-attachment:fixed;

}
.categories li a{
        color: #666;
        background-color: #AAA;
}
.categories .selected a {
        background-color: #FFF;
        border-bottom: 3px solid #FFF;
}
.categories li a:hover {
        color: #000;
        background-color: #DDD;
}

#main {

        background-position:center;
        background-repeat:no-repeat;
}
h1 {
        color: #666;
}
h2 {
        color: #222;
}
.listbox li:hover {
        background: #DFDFDF;
        color: #222;
}
```

1

```
.message {
        border: 1px solid red;
        background: #FFFFFF;
}
.error {
        background-color:
#BF0000!important;
}
.warning {
        background-color:
#BF6E00!important;
}
.info {
        background-color:
#6EBF00!important;
}
.notice {
        background-color:
#6EBF00!important;
}
.errcont {
        border: 3px solid #F99;
        background-color: #FEE;
        margin: 2px;
}
.errfield {
        background-color: #FCC;
        color: #F00;
}
.toolbar {
        #background: #BCBCBC;
}
.copyright, .copyright a {
        color: #666;
}
.logout {
        background-color: white;
}
.dbg_button {
        color: #F66;
        background-color: #DFDFDF;
}
```

2

```
#list_frm{
-moz-border-radius:10px;
border-radius:10px;
margin:0px;
padding:10px;
background: url(/images/panel2.png);
}
#Puerta_frm{
-moz-border-radius:10px;
border-radius:10px;
margin:0px;
padding:10px;
background: url(/images/panel2.png);
}
#CameraShow_frm{
-moz-border-radius:10px;
border-radius:10px;
margin:0px;
padding:10px;
background: url(/images/panel2.png);
}
#SetSystemState_frm{
-moz-border-radius:10px;
border-radius:10px;
margin:0px;
padding:10px;
background: url(/images/panel2.png);
}
#manageSystemServices_frm{
-moz-border-radius:10px;
border-radius:10px;
margin:0px;
padding:10px;
background: url(/images/panel2.png);
}
#listControlPorts_frm{
-moz-border-radius:10px;
border-radius:10px;
margin:0px;
padding:10px;
background: url(/images/panel2.png);
}
```

3

```
#configureInterface_frm{
-moz-border-radius:10px;
border-radius:10px;
margin:0px;
padding:10px;
background: url(/images/panel2.png);
}
#addControlDevice_frm{
-moz-border-radius:10px;
border-radius:10px;
margin:0px;
padding:10px;
background: url(/images/panel2.png);
}
#configureControlPorts_frm{
-moz-border-radius:10px;
border-radius:10px;
margin:0px;
padding:10px;
background: url(/images/panel2.png);
}
#emailSettings_frm{
-moz-border-radius:10px;
border-radius:10px;
margin:0px;
padding:10px;
background: url(/images/panel2.png);
}
#installPackage_frm{
-moz-border-radius:10px;
border-radius:10px;
margin:0px;
padding:10px;
background: url(/images/panel2.png);
}
#addUser_frm{
-moz-border-radius:10px;
border-radius:10px;
margin:0px;
padding:10px;
background: url(/images/panel2.png);
}
```

4

```
#addUser_frm{
-moz-border-radius:10px;
border-radius:10px;
margin:0px;
padding:10px;
background: url(/images/panel2.png);
}
#delUser_frm{
-moz-border-radius:10px;
border-radius:10px;
margin:0px;
padding:10px;
background: url(/images/panel2.png);
}
#addIpcam_frm{
-moz-border-radius:10px;
border-radius:10px;
margin:0px;
padding:10px;
background: url(/images/panel2.png);
}
#delIpcam_frm{
-moz-border-radius:10px;
border-radius:10px;
margin:0px;
padding:10px;
background: url(/images/panel2.png);
}
#manageCameras_frm{
-moz-border-radius:10px;
border-radius:10px;
margin:0px;
padding:10px;
background: url(/images/panel2.png);
}
```

5

```
#modUser_frm{
-moz-border-radius:10px;
border-radius:10px;
margin:0px;
padding:10px;
background: url(/images/panel2.png);
}
#addAgentToNetwork_frm{
-moz-border-radius:10px;
border-radius:10px;
margin:0px;
padding:10px;
background: url(/images/panel2.png);
}
.comments{
-moz-border-radius:10px;
border-radius:10px;
margin:0px;
padding:10px;
background: url(/images/panel2.png);
}
#browseFiles_frm{
-moz-border-radius:10px;
border-radius:10px;
margin:0px;
padding:10px;
background: url(/images/panel2.png);
}
#showEvents_frm{
-moz-border-radius:10px;
border-radius:10px;
margin:0px;
padding:10px;
background: url(/images/panel2.png);
}
```

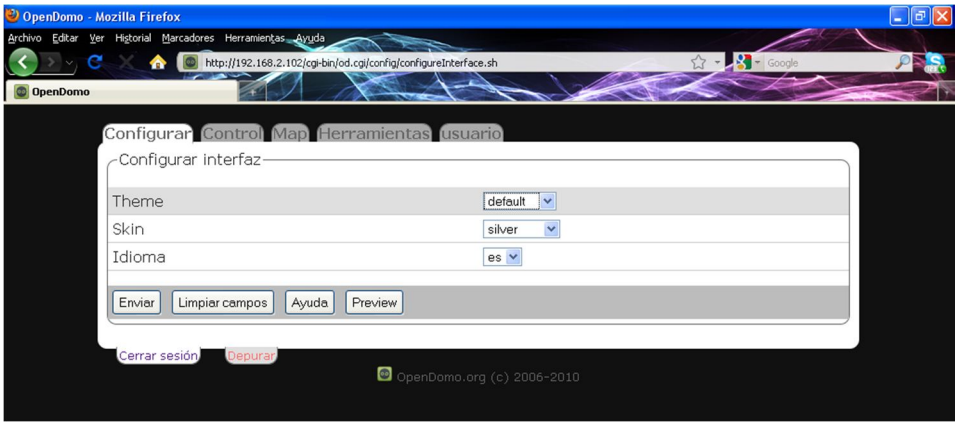
6

Elaboración: Ricardo Almendares

Nota: Las imágenes deben ser guardadas en la siguiente dirección: /var/www/images y el script debe ser colocado dentro de una carpeta en la ruta /var/www/skins

Una vez creados nuestros scripts procedemos a seleccionarlos mediante el menú **configurar > configurar interfaz**, escogemos el tema y skin que hicimos y le damos clic en enviar

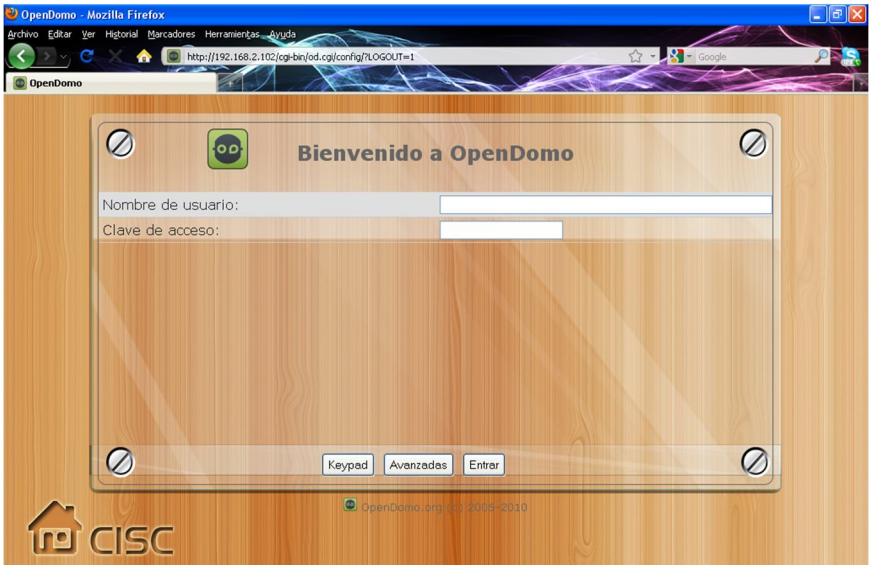
Gráfico 2 Selección de interfaz



Elaboración: Ricardo Almendares

A continuación podremos observar la interfaz seleccionada

Gráfico 3 Salida de nueva interfaz



Elaboración: Ricardo Almendares

Detalles Técnicos adicionales del Hardware

Para obtener más información sobre la tarjeta Arduino véase en la documentación principal el anexo A.

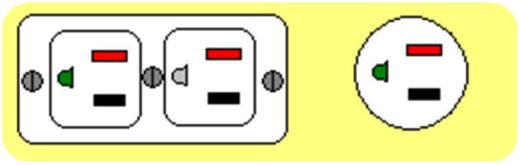
A continuación veremos la parte eléctrica y electrónica que se utilizo para el desarrollo de este proyecto.

Toma Polarizada

Los tomacorrientes se denominan como polarizados y no polarizados, estos son los más utilizados en una casa normal, aunque para proteger todos los aparatos conectados lo ideal es que se coloquen tomacorrientes polarizados.

Tomacorriente polarizado: Este tomacorriente se caracteriza por tener tres puntos de conexión, el vivo o positivo, el negativo y el de tierra física, es muy importante el uso de estos tomacorrientes.

Gráfico 4 Tomacorriente Polarizado

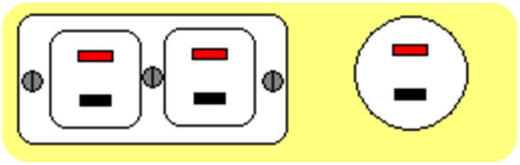


Fuente: http://www.electricidadbasica.net/inst_tomacorriente.htm

Tomacorriente no polarizado: Este tomacorriente únicamente tiene 2 puntos de conexión, el vivo o positivo y el negativo; este tipo de tomacorriente no es recomendable

para aparatos que necesiten una protección adecuada contra sobrecargas y descargas atmosféricas.

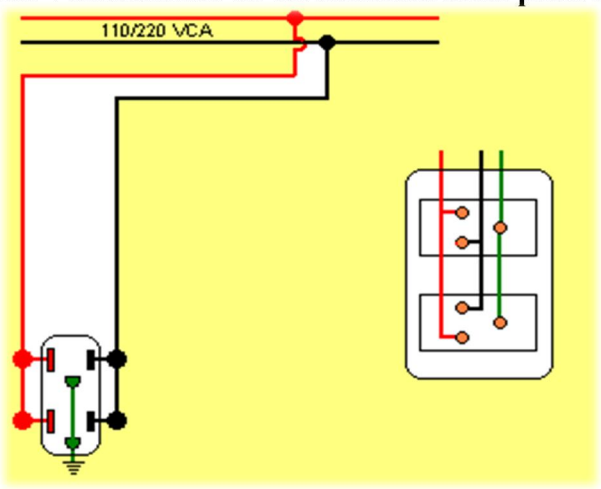
Gráfico 6 Tomacorriente no Polarizado



Fuente: http://www.electricidadbasica.net/inst_tomacorriente.htm

Para la instalación de un tomacorriente se debe de desmontar la toma anterior quitando los tornillos que aseguran el tomacorriente a la caja, luego, aflojar los tornillos que aseguran los cables y colocar el nuevo. Si es una instalación nueva, primero debemos de colocar los cables dentro del tubo. En el caso de los tomacorrientes los cables se conectan al positivo y negativo de la instalación directamente.

Gráfico 7 Instalación de un tomacorriente polarizado



Fuente: http://www.electricidadbasica.net/inst_tomacorriente.htm

En el gráfico puede verse que debemos de conectar tres cables para instalar un tomacorriente polarizado:

ROJO: Este debe de conectarse a la línea viva o positiva de la instalación eléctrica. **NEGRO**: Este debe de conectarse a la línea negativa de la instalación eléctrica. **VERDE**: Este corresponde a la tierra física instalación eléctrica.

Introducción a los Servomotores

Gráfico 8 Servomotor de rotación continua

Fuente: http://www.drivecontroldetails.info/images/servo-motor-position.jpg

Un servomotor es un motor de corriente continua que tiene la capacidad de ser controlado en posición. Es capaz de ubicarse en cualquier posición dentro de un rango de operación y mantenerse estable en dicha posición. Los servomotores se suelen utilizar en robótica, automática y modelismo (vehículos por radio-control, RC) debido a su gran precisión en el posicionamiento.

En general, los servomotores suelen estar compuestos por 4 elementos fundamentales:

- **Motor de corriente continua (DC):** es el elemento que le brinda movilidad al servomotor. Cuando se aplica un potencial a sus dos terminales, el motor gira en un sentido a su velocidad máxima. Si el voltaje aplicado sus dos terminales es inverso, el sentido de giro también se invierte.
- **Engranajes reductores:** Tren de engranajes que se encarga de reducir la alta velocidad de giro del motor para acrecentar su capacidad de torque.
- **Sensor de desplazamiento:** Suele ser un potenciómetro colocado en el eje de salida del servomotor que se utiliza para conocer la posición angular del motor.
- **Circuito de control:** Es una placa electrónica que implementa una estrategia de control de la posición por realimentación. Para ello, este circuito compara la señal de entrada de referencia (posición deseada) con la posición actual medida por el potenciómetro. La diferencia entre la posición actual y la deseada es amplificada y utilizada para mover el motor en la dirección necesaria para reducir el error.

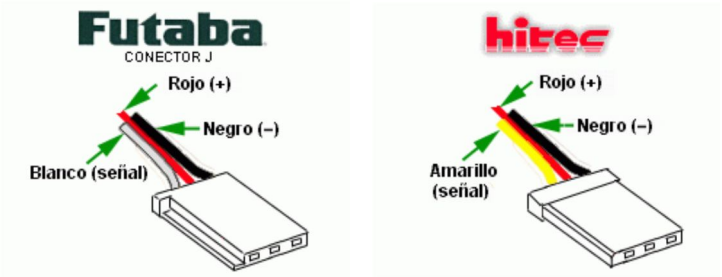
Gráfico 9 Componentes de un servomotor

Fuente: http://www.aurova.ua.es:8080/proyectos/dpi2005/docs/publicaciones/pub09-ServoMotores/servos.pdf

Principios de Funcionamiento

Los servomotores disponen de tres cables: dos cables de alimentación (positivo y negativo/ground) que suministran un voltaje 4.8-6v y un cable de control que indica la posición deseada al circuito de control mediante señales PWM (Pulse Width Modulation).

Gráfico 10 Colores de los cables de los principales fabricantes de servomotores



Fuente: http://www.aurova.ua.es:8080/proyectos/dpi2005/docs/publicaciones/pub09-ServoMotores/servos.pdf

Las señales PWM utilizadas para controlar los servomotores están formadas por pulsos positivos cuya duración es proporcional a la posición deseada del servomotor y que se repiten cada 20 ms (50Hz). Todos los servos pueden funcionar correctamente en un rango de movimiento de 90º, que se corresponde con pulsos PWM comprendidos entre 0.9 y 2.1 ms. Sin embargo, también existen servomotores que se pueden mover en un rango extendido de 180º y sus pulsos de control varían entre 0.5 y 2.5 ms. Antes de utilizar un servomotor habrá que comprobar experimentalmente su rango de movimiento para no dañarlo. Para mantener fijo un servomotor en una posición habrá que enviar

periódicamente el pulso correspondiente; ya que si no recibe señales, el eje del servomotor quedará libre y se podrá mover ejerciendo una leve presión.

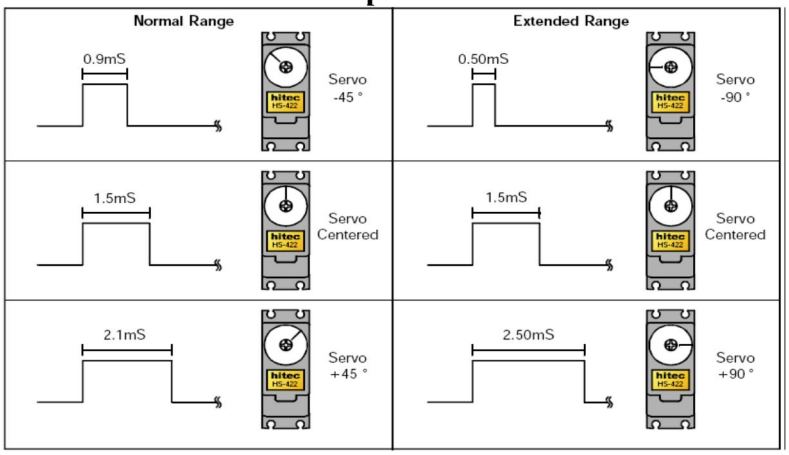
La conexión de un servomotor puede variar según el fabricante, a continuación algunas configuraciones para marcas conocidas.

Cuadro 6 Configuración de Servomotores para marcas conocidas

Fabricante	Voltaje positivo	Tierra	Señal de control
Futaba	Rojo	Negro	Blanco
Dong Yang	Rojo	Marrón	Naranja
Hobico	Rojo	Negro	Amarillo
Hitec	Rojo	Negro	Amarillo
JR	Rojo	Marrón	Naranja
Airtronics	Rojo	Negro	Naranja
Fleet	Rojo	Negro	Blanco
Krafr	Rojo	Negro	Naranja
E-Sky	Rojo	Negro	Blanco

Fuente: http://www.olimex.cl/tutorial/Guia%20MCI%20-%20Servo%20motor%20con%20Arduino.pdf

Gráfico 11 Pulsos PWM para controlar servomotores



Fuente:http://www.aurova.ua.es:8080/proyectos/dpi2005/docs/publicaciones/pub09-ServoMotores/servos.pdf

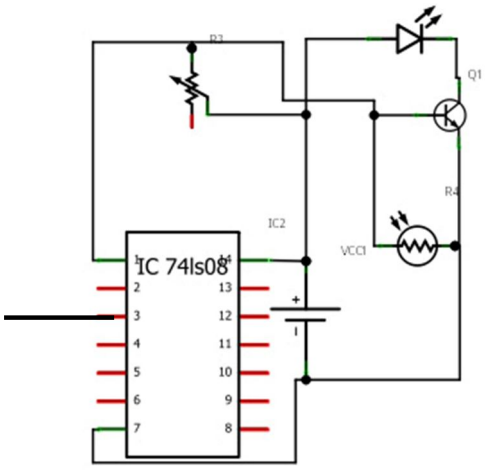
Alarma Láser

Para desarrollar esta alarma se necesitó los siguientes materiales:

- Transistor NPN 2n3904
- LDR (Light Dependent Resistor, o Resistor Dependiente de la Luz)
- Potenciómetro de 1Mega Ohmio
- Circuito 74LS08 (cuatro puertas AND de dos entradas)
- Puntero Laser
- Fuente de voltaje de 5v (para el proyecto se utilizó la fuente de voltaje de la tarjeta Arduino)
- LED(Light Emitting Diode, o Diodo Emisor de Luz)

A continuación el esquema utilizado en el proyecto:

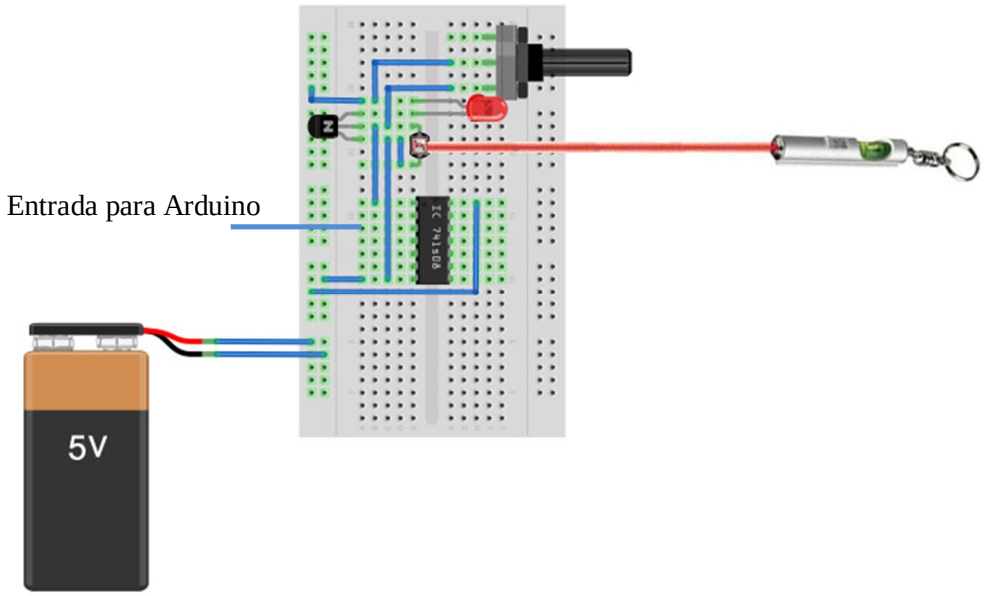
Gráfico 12 Esquema de la alarma laser



Fuente: Ricardo Almendares

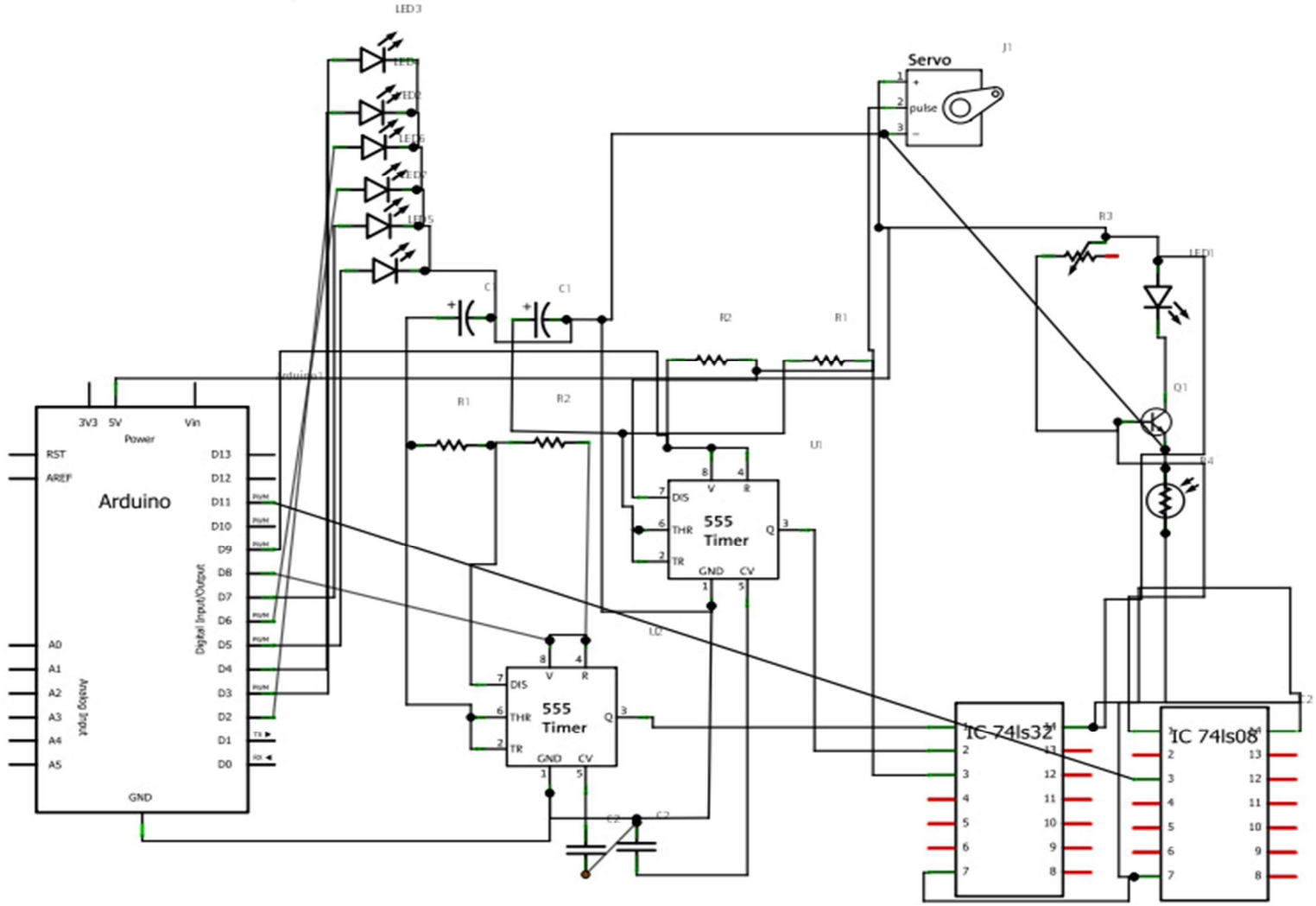
En esta alarma hay un haz láser que está en contacto permanente con la LDR, al perder el contacto el transistor entra en estado de saturación y se encenderá el LED esto quiere decir que ha ingresado una persona (intruso). Dependiendo del transistor utilizado y de la LDR se deberá recalcular la resistencia de base, el potenciómetro nos permitirá regular la intensidad a que queramos que se encienda la alarma. El circuito 74LS08 nos servirá para obtener 5v los cuales nos servirán como entrada en la tarjeta Arduino

Gráfico 13 Montaje de la alarma en un protoboard



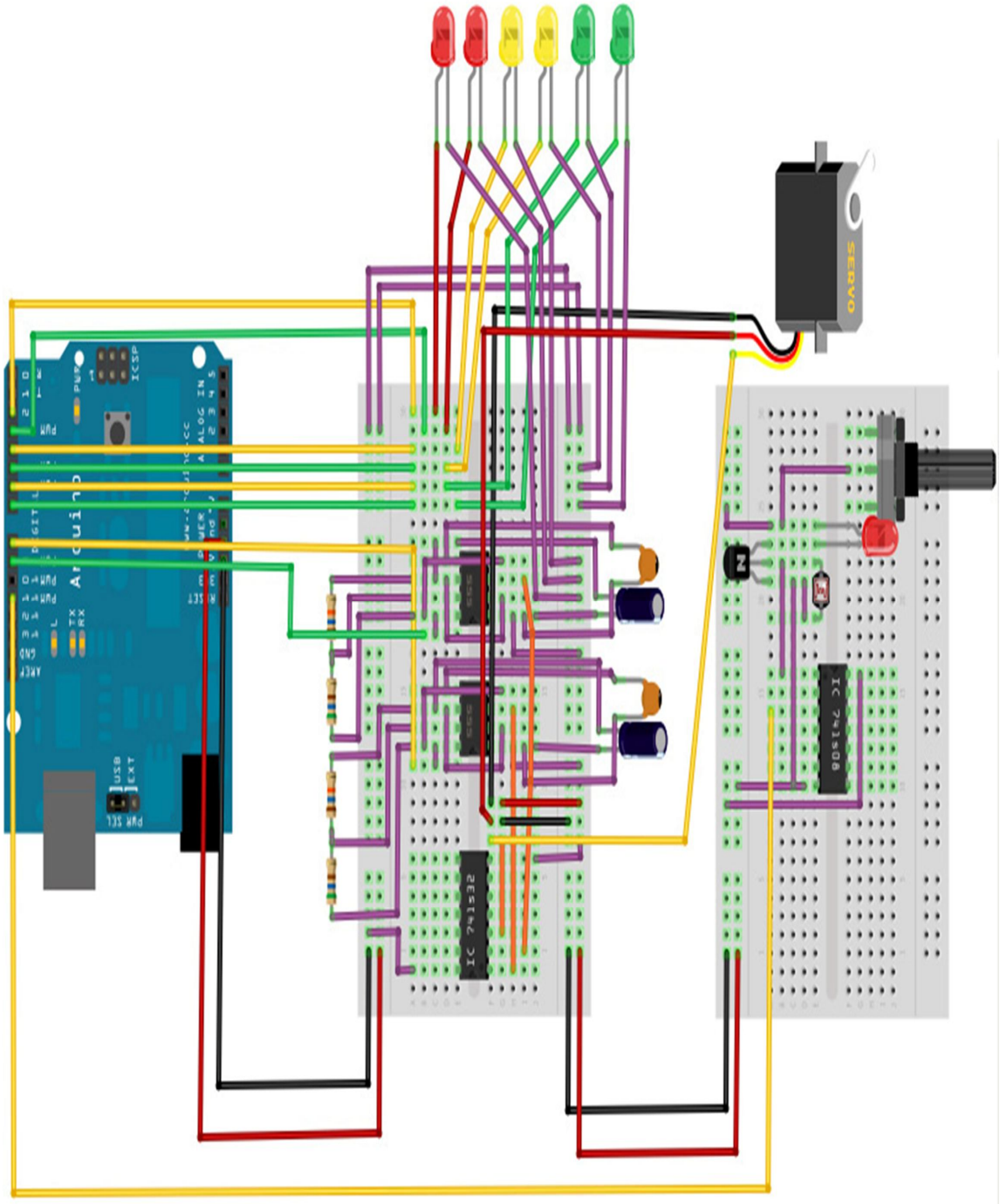
Fuente: Ricardo Almendares

Gráfico 14 Esquema del tablero principal



Fuente: Ricardo Almendares

Gráfico 15 Montaje del tablero principal



Fuente: Ricardo Almendares

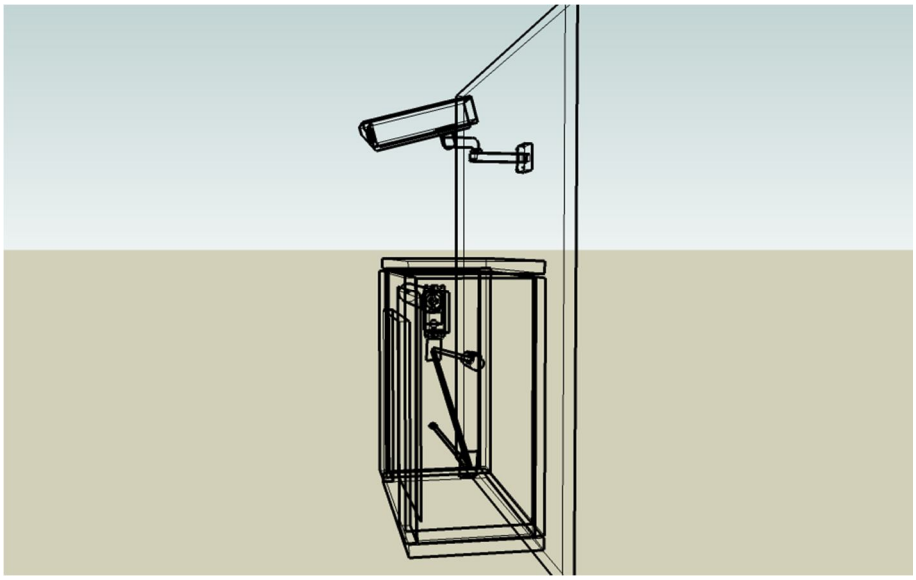
Los 6 LEDs indican las 6 diferentes áreas de nuestra maqueta las cuales podemos controlar desde el sistema OpenDomo.

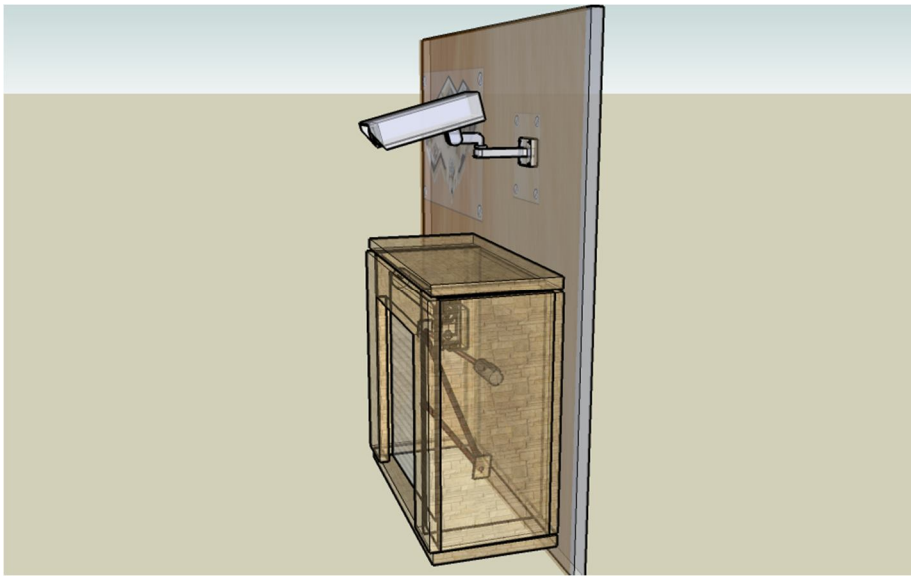
Gráfico 16 Áreas de la maqueta

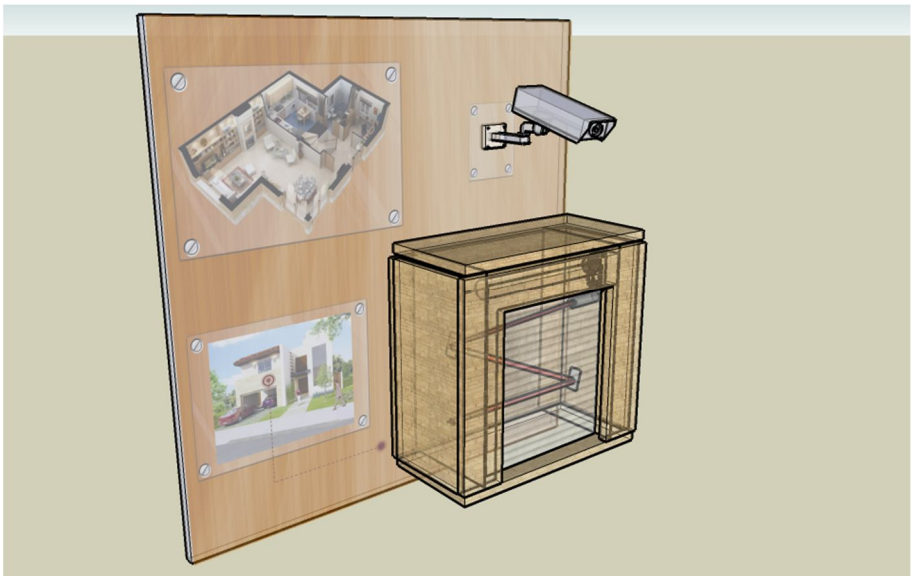
Fuente: http://2.bp.blogspot.com/_YKw_c7D0WkU/Sw6udKUtj7I/AAAAAAAAAC8/oR4TsQkF-YE/s1600/Seccion+planta+baja.jpg

Como último paso para este proyecto se realizó una maqueta para poder demostrar las múltiples ventajas y facilidades que ofrece el uso del sistema OpenDomo, proporcionando así una solución rentable y flexible para aplicaciones de seguridad y supervisión remota.

Gráfico 17 Diferentes perspectivas del prototipo







Fuente: Ricardo Almendares
Elaboración: Ricardo Almendares